AMENDMENTS TO THE CLAIMS

Please cancel claims 10 and 18 without prejudice. Kindly amend claims 1, 19-20, 24, 27 and 38 as shown in the following listing of claims. The listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims

1. (currently amended) A cache memory, comprising:

a plurality of storage elements, arranged as a last-in-first-out (LIFO) memory, for storing data exclusively specified by push instructions, wherein each of said push instructions implicitly specifies a data memory address based on a value stored in a microprocessor stack pointer register rather than explicitly specified by the push instruction, said LIFO memory having a top one of said plurality of storage elements for storing a cache line of data specified by a plurality of most-recent said push instructions, and for storing a virtual address and a physical address of said cache line;

a comparator, coupled to said top storage element, for comparing said cache line virtual address with a source virtual address of data requested by a load instruction; and

an output, coupled to said comparator, for speculatively indicating said data requested by said load instruction is present in the cache memory if said comparator indicates said source virtual address matches said cache line virtual address stored in said top storage element, prior to determining whether a source physical address of said data requested by said load instruction matches said physical cache line address.

2. (original) The cache memory of claim 1, wherein said source physical address and said physical cache line address each comprise an upper portion of translated address bits and a lower portion of untranslated address bits.

3. (original) The cache memory of claim 2, wherein said translated address bits comprise an address of a physical memory page, wherein said untranslated address bits comprise an offset within said physical memory page.

4. (original) The cache memory of claim 2, wherein said source virtual address comprises said untranslated lower portion of said source physical address.

5. (original) The cache memory of claim 4, wherein said source virtual address further comprises an upper portion of untranslated address bits appended to said untranslated lower portion.

6. (original) The cache memory of claim 5, wherein said upper portion of untranslated address bits of said source virtual address comprises an address of a virtual memory page.

7. (original) The cache memory of claim 2, wherein said source virtual address and said virtual cache line address comprise said untranslated lower portion of said source physical address and said physical cache line address, respectively, wherein said source virtual address and said virtual cache line address each further comprise an upper portion of hashed untranslated address bits appended to said untranslated lower portion.

8. (original) The cache memory of claim 1, wherein said load instruction comprises an instruction that explicitly specifies said source virtual address.

9. (original) The cache memory of claim 1, wherein said load instruction comprises an instruction that does not implicitly specify said source virtual address relative to a stack pointer register value.

10. (canceled)

11. (original) The cache memory of claim 1, further comprising:

a second comparator, coupled to said plurality of storage elements, for comparing said source physical address with said physical cache line address.

12. (original) The cache memory of claim 11, further comprising:

a second output, coupled to said second comparator, for indicating said first output incorrectly indicated that said data requested by said load instruction is present in the cache memory, if said second comparator indicates said source physical address does not match said physical cache line address subsequent to said first output indicating said data requested by said load instruction is present in the cache memory.

13. (original) The cache memory of claim 1, further comprising:

a second comparator, coupled to said plurality of storage elements, for comparing said source virtual address with a virtual address of a cache line stored in a next-to-top one of said plurality of storage elements, said next-to-top storage element storing a cache line of data specified by a plurality of next-most-recent push instructions to said plurality of most-recent push instructions.

14. (original) The cache memory of claim 13, wherein said first output indicates said data requested by said load instruction is present in the cache memory if said second comparator indicates said source virtual address matches said cache line virtual address stored in said next-to-top storage element, prior to determining whether said source physical address matches said physical cache line address of said cache line stored in said next-to-top storage element.

15. (original) The cache memory of claim 1, further comprising:

a plurality of comparators, coupled to said plurality of storage elements, for comparing said source physical address with a plurality of physical cache line addresses stored in said plurality of storage elements.

16. (original) The cache memory of claim 15, further comprising:

a second output, coupled to said plurality of comparators, for indicating said data requested by said load instruction is non-speculatively present in the cache memory if said comparator indicates said source virtual address does not match said cache line virtual address stored in said top storage element, and said plurality of comparators indicates said source physical address matches one of said plurality of physical cache line addresses stored in said plurality of storage elements.

17. (original) The cache memory of claim 16, wherein said second output indicates said data requested by said load instruction is non-speculatively present in the cache memory in a second clock cycle subsequent to a first clock cycle in which said output indicates said data requested by said load instruction is not present in the cache memory.

18. (canceled)

19. (currently amended) The cache memory of claim 1, wherein a computer program product comprising a computer usable medium having computer readable program code ~~causes~~ provides the cache memory, wherein said computer program product is for use with a computing device.

20. (currently amended) A microprocessor, comprising:

a first level-one cache memory, for caching data exclusively specified by push instructions, said first cache memory comprising a last-in-first-out (LIFO) stack memory having a top entry for storing a cache line of data associated with newest push instruction data;

a second level-one cache memory, for caching data exclusively specified by non-push memory access instructions, said second cache memory comprising a non-LIFO memory; and

control logic, coupled to said first and second cache memories, for causing said first cache memory to speculatively provide from said top entry data specified by a load instruction, if a virtual address specified by said load instruction matches a virtual address of said cache line stored in said top entry.

21. (original) The microprocessor of claim 20, wherein if said virtual address specified by said load instruction does not match said virtual address of said cache line stored in said top entry, but if a physical address translated from said virtual address specified by said load instruction matches a physical address of one of a plurality of cache lines stored in said first cache memory, then said control logic causes said first cache memory to non-speculatively provide said data specified by said load instruction from said matching one of said plurality of cache lines.

22. (original) The microprocessor of claim 21, wherein if said physical address translated from said virtual address specified by said load instruction does not match said physical address of any of said plurality of cache lines stored in said first cache memory, said control logic causes said second cache memory to non-

speculatively provide said data specified by said load instruction if said physical address translated from said virtual address specified by said load instruction matches a physical address of a cache line stored in said second cache memory.

23. (original) The microprocessor of claim 20, further comprising:

a plurality of physical address comparators, coupled to said control logic, for detecting a condition in which said control logic incorrectly caused said first cache memory to speculatively provide from said top entry data specified by said load instruction.

24. (currently amended) The microprocessor of claim 23, wherein said condition is detected based on a determination that a physical address translated from said virtual address specified by said load instruction misses in said first cache memory.

25. (original) The microprocessor of claim 23, further comprising:

a microcode memory, coupled to said control logic, for storing microcode instructions for recovering from said condition.

26. (original) The microprocessor of claim 20, wherein said control logic causes said first cache memory to speculatively provide from said top entry said data specified by said load instruction if said virtual address specified by said load instruction matches said virtual address of said cache line stored in said top entry, prior to determining whether a physical address translated from said virtual address specified by said load instruction matches a physical address of said cache line stored in said top entry.

27. (currently amended) A method for performing a speculative load operation from a stack memory cache, the method comprising:

storing stack memory data into a cache memory in a last-in-first-out (LIFO) manner;

determining whether a virtual address of a load instruction matches a virtual address of data stored in a top entry of the cache memory, after said storing;

determining whether a physical address of the load instruction matches a physical address of the data stored in the top entry; and

speculatively providing the data from the top entry if the virtual address of the load instruction matches the virtual address of the data stored in the top entry, but before said determining whether the physical address of the load instruction matches the physical address of the data stored in the top entry.

28. (original) The method of claim 27, further comprising:

translating the physical address of the load instruction from the virtual address of the load instruction, prior to said determining whether the physical address of the load instruction matches the physical address of the data stored in the top entry.

29. (original) The method of claim 28, wherein said translating the physical address of the load instruction from the virtual address of the load instruction is performed substantially in parallel with said determining whether the virtual address of the load instruction matches the virtual address of data stored in the top entry of the cache memory.

30. (original) The method of claim 28, wherein said translating the physical address of the load instruction from the virtual address of the load instruction is performed by a translation lookaside buffer.

31. (original) The method of claim 28, wherein said translating the physical address of the load instruction from the virtual address of the load instruction comprises translating a virtual memory page address to a physical memory page address.

32. (original) The method of claim 27, further comprising:

generating an exception signal, after said providing the data from the top entry if the virtual address of the load instruction matches the virtual address of the data stored in the top entry, if the physical address of the load instruction does not match the physical address of the data stored in the top entry.

33. (original) The method of claim 32, wherein the exception signal indicates the data provided to the load instruction from the top entry was incorrect data.

34. (original) The method of claim 32, further comprising:

providing correct data to the load instruction in response to said exception signal.

35. (original) The method of claim 34, wherein said providing correct data to the load instruction in response to said exception signal comprises a microprocessor comprising the cache memory executing a microcode routine to provide the correct data.

36. (original) The method of claim 27, further comprising:

storing non-stack memory data into a second cache memory into locations of the second cache memory based on an address of the non-stack memory data; and

determining whether the physical address of the load instruction matches a physical address of data stored in the second cache memory; and

providing the data from the second cache memory if the physical address of the load instruction matches a physical address of data stored in the second cache memory.

37. (original) The method of claim 36, wherein said determining whether the physical address of the load instruction matches a physical address of data stored in the second cache memory is performed substantially in parallel with said determining whether the physical address of the load instruction matches the physical address of the data stored in the top entry of the LIFO cache memory.

38. (currently amended) A computer ~~data signal~~ program product embodied ~~in~~ on a ~~transmission~~ computer-readable medium, comprising:

computer-readable program code for providing a cache memory, said program code comprising:

first program code for providing a plurality of storage elements, arranged as a last-in-first-out (LIFO) memory, for storing data exclusively specified by push instructions, wherein a push instruction is an instruction that moves data to memory, wherein a memory address of the data is implicitly specified by the push instruction based on a value stored in a microprocessor stack pointer register rather than explicitly by the push instruction, said LIFO memory having a top one of said plurality of storage elements for storing a cache line of data specified by a plurality of most-recent said push instructions, and for storing a virtual address and a physical address of said cache line;

second program code for providing a comparator, coupled to said top storage element, for comparing said cache line virtual address with a source virtual address of data requested by a load instruction; and

third program code for providing an output, coupled to said comparator, for speculatively indicating said data requested by said load instruction is present in the cache memory if said comparator indicates said source virtual address matches said cache line virtual address stored in said top storage element, prior to determining whether a source physical address of said data requested by said load instruction matches said physical cache line address.